

RESEARCH

Open Access

Non-volatile memory reduction based on 1-D memory space mapping of a specific set of QC-LDPC codes

Chung-Ping Young¹, Chung-Chu Chia¹, Chao-Chin Yang^{2*} and Chung-Ming Huang³

Abstract

Supporting a great diversity of multi-rate \mathbf{H} -matrices for multiple communication protocols requires a large amount of non-volatile memory, which may consume a large silicon area or logic elements and constrain the implementation of an overall decoder. Therefore, schemes for memory reduction are necessary to make the parity-check storage more compact. This study proposes a specific set of quasi-cyclic low-density parity-check (LDPC) (QC-LDPC) codes which can transfer a traditional two-dimensional (2-D) parity-check matrix (\mathbf{H} -matrix) into a one-dimensional (1-D) memory space. Compared to the existing schemes, the proposed codes and memory reduction scheme do achieve significant reduction rates. Within a fixed memory space, many more \mathbf{H} -matrices for diverse communication protocols can be saved via the proposed QC-LDPC codes, which are well constructed from modified Welch-Costas sequences. Furthermore, relatively good error performances, which outperform computer-generated random LDPC codes and Sridhara-Fuja-Tanner codes, are also shown in our simulation results. Consequently, we conclude that the proposed QC-LDPC codes can enlarge the capacity for saving much more low-BER (bit error rate) \mathbf{H} -matrices within a fixed memory space.

1. Introduction

Low-density parity-check (LDPC) codes were first introduced by Gallager in 1962, but they were rarely used since implementing them in hardware was impractical in the 1960s. The value of LDPC codes was rediscovered by Mackay and Neal in 1996 [1]. Since then, LDPC codes have gained a lot of attention due to their excellent error correction capability. A binary (j, k) -regular LDPC code is defined as the null space of a sparse parity-check matrix \mathbf{H} over $\text{GF}(2)$ and satisfies the following properties: (1) each column has weight j ; (2) each row has weight k ; (3) no two rows (or two columns) have more than one 1-component in common; (4) both j and k are much smaller than the code length.

Most methods for designing good LDPC codes are based on random constructions, but the lack of structure makes the encoding process complicated. Furthermore, the non-volatile memory required to store the parity-check matrices may be prohibitive in practical applications.

Nowadays, some wireless devices are designed to be both tiny and capable of supporting multiple communication functions, such as WLAN [2,3], 3G, DVB-S2 [4], CMMB [5], etc. Therefore, a great diversity of LDPC codes is employed for the demand of error corrections. Compared to overall decoders, the storage for multiple LDPC \mathbf{H} -matrices is very area consuming. Therefore, memory reduction schemes are necessary for reducing the memory requirements as much as possible. quasi-cyclic LDPC (QC-LDPC) codes are always employed for this purpose since even trivial approaches can achieve huge gains in memory reduction.

There are two primary types of parity-check matrices of LDPC codes: the pseudorandom matrix [6] and the quasi-cyclic matrix [7,8]. The latter, whose encoding complexity is directly proportional to the code length, is widely applied in consumer electronics. Several classes of QC-LDPC codes [8-10] have been proposed. Such codes can achieve good error performance comparable with computer-generated random LDPC codes. However, in terms of implementation aspects, QC-LDPC codes for multi-rate communication sessions and diverse communication protocols need to be stored concurrently. As indicated in

* Correspondence: ksuee13@gmail.com

²Department of Electro-Optical Engineering, Kun Shan University, Tainan 710, Taiwan, Republic of China

Full list of author information is available at the end of the article

[6], directly employing the lookup tables for saving multiple \mathbf{H} -matrices is always prohibitive.

The motivation of this study is to propose a specific set of QC-LDPC codes with extremely low memory requirements. To achieve this goal, we introduce properly constructed QC-LDPC codes which can be classified as a specific set of formerly proposed modified Welch-Costas (MWC-OCS) codes [11]. These LDPC codes are constructed by multilevel sequences [8,12,13] with the property that any two different rows have at most one element in common. Based on our proposed 2-D to 1-D memory space mapping, each code in the specific set can achieve a huge reduction rate due to its particular structure.

Some studies related to \mathbf{H}_{ROM} storage are shown in Table 1, in which critical implementation issues for LDPC codes are presented:

(1) As indicated in [14], the unreduced \mathbf{H}_{ROM} is definitely area-consuming. In fact, it may occupy 60% area of the overall decoder. Therefore, reduction schemes are necessary for degrading the memory demand.

(2) The reduction impact is still unobvious in some LDPC codes with irregular structures. Possible approaches have been applied in [15].

(3) From [5], some code structures can massively be reduced to merely 8.5% of the overall area. This achievement is meaningful, especially when the target device is specified. For example, after optimization, this decoder can easily fit an ALTERA EP2S15F484C3 FPGA (250 K gates), and hence the search for a larger or more expensive device is no longer necessary.

(4) The study [2] indicates that supporting multiple \mathbf{H} -matrices will increase the memory demand. Although the required non-volatile memory has been optimized as a $[324 \times 48]$ low-power ROM (60 K gates), it still consumes a large portion of the hardware resource.

To address the critical issues mentioned above, we propose extremely compact codes to significantly reduce a single \mathbf{H} -matrix storage. Moreover, a reduction scheme via data mapping and the proposed codes are also demonstrated to further degrade the memory demand for multi-rate \mathbf{H} -matrices storage. Both of the proposed codes and the further reduction scheme can achieve huge reduction rates. Compared to existing approaches, this study can squeeze many more \mathbf{H} -matrices within a fixed memory space.

This article is organized as follows. In the introduction section, we highlighted the need for \mathbf{H}_{ROM} reduction and clarify the motivation of our approach. In Section 2, we explain how a QC-LDPC parity-check matrix is characterized by merely storing the shift values of its identity sub-matrices for memory efficiency. Our similar codes with extremely low memory requirements for CMMB, WLAN, and WIMAX are introduced in Section 3. The proposed MWC-OCS LDPC block codes and their 1-D memory space mapping are introduced in Sections 4 and 5. In Section 6, a design example demonstrates how multi-rate \mathbf{H} -matrices storage can be further reduced. For a tiny device designed to support more \mathbf{H} -matrices for diverse communication protocols, this example should not be considered negligible. Finally, we offer conclusions in Section 7.

2. QC-LDPC block codes storage

For QC-LDPC codes, storing \mathbf{H} [2] involves saving the shift values of identity sub-matrices and their column positions. The column positions can easily be generated by an address generator unit (AGU) [5]. As for the storage of shift values, it requires a 2-D matrix which can only be carried out by a non-volatile memory space. Non-volatile memory severely consumes the logic elements. Properly constructed LDPC codes require less non-volatile memory and can make the hardware resource available for decoder improvements or more \mathbf{H} -matrices storage.

The content stored in the non-volatile memory is used for generating actual addresses which point to soft messages stored in the volatile memory (RAM). An actual address can easily be determined by adding an offset address to the base address. A decoder retrieves a soft message by accessing the RAM via an actual memory address. The number of addresses required to retrieve all the content of message RAM equals to the number of 1-components in the \mathbf{H} -matrix. Each 1-component in \mathbf{H} represents a RAM address. All the RAM addresses needed for the decoder to obtain the soft messages are stored in the non-volatile memory. Without optimization, the size of this non-volatile memory is equal to $Z \times U$ bits, where Z is the address width ($Z \geq \log_2 N$) [5], and U is the total number of 1-components in an \mathbf{H} -matrix with code length N . In QC-LDPC block codes, the non-volatile memory for recording the RAM addresses can be replaced by a reduced 2-D \mathbf{Y} -matrix, in which only shift values are

Table 1 The area-consuming feature of the \mathbf{H}_{ROM}

Works	\mathbf{H}_{ROM} reduction	\mathbf{H}_{ROM} (gates)	Overall decoder (gates)	\mathbf{H}_{ROM} /decoder ratio (%)	\mathbf{H}_{ROM} /EP2S15F484C3 ratio (%)
[14]	No	1180 K	1940 K	60.8	472
[15]	Yes	367 K	837 K	43.8	147
[5]	No	N.A	294 K	N.A	> 100
	Yes	7 K	82 K	8.5	2.8
[2]	Yes	60 K	N.A	N.A	24

stored. With the content of \mathbf{Y} and an AGU [5], the original memory space can effectively be reduced. This section shows how an \mathbf{H} -matrix of a QC-LDPC code can be compactly stored and how actual addresses can be determined by an AGU.

2.1. Non-zero elements in hardware implementation

In terms of hardware implementations, the position of '1' in the same column of a parity-check matrix actually represents an offset address. An actual address, which points to a soft message stored in the volatile memory, can be generated by the addition of an offset address and a base address. For an N -bit codeword, the offset addresses range between 0 and $N - 1$. The actual addresses can be generated on the fly by an AGU which spans all the required memory addresses for message retrieval. The AGU requires only a small amount of data which characterizes the feature of \mathbf{H} . This required data varies with different code structures. Chose the smallest integer Z ($Z \geq \log_2 N$), then each non-zero element in the \mathbf{H} actually denotes to a Z -bit offset address which can be used to retrieve a soft message stored in the corresponding message memory.

2.2. Message address determination

In Figure 1, a (3, 4)-regular QC-LDPC code is composed of $12 \times p \times p$ ($p = 3$) sub-matrices and can be expressed as a (12, 3, 4) QC-LDPC code with codeword length $N = 12$. An offset address represented by $1_{r,v}$ is located at the r th ($0 \leq r \leq jp-1$) row of the parity check \mathbf{H} -matrix and v th ($0 \leq v \leq k-1$) column in which k circulant permutation sub-matrices are placed. A sub-matrix located at the u th row and v th column is cyclically shifted by $y_{u,v}$ positions,

$1_{0,0}$				$1_{0,1}$			$1_{0,2}$		$1_{0,3}$		
	$1_{1,0}$				$1_{1,1}$			$1_{1,2}$		$1_{1,3}$	
		$1_{2,0}$	$1_{2,1}$			$1_{2,2}$					$1_{2,3}$
				$1_{3,1}$	$1_{3,2}$						$1_{3,3}$
		$1_{4,0}$			$1_{4,1}$	$1_{4,2}$					$1_{4,3}$
$1_{5,0}$			$1_{5,1}$				$1_{5,2}$	$1_{5,3}$			
	$1_{6,0}$	$1_{6,1}$				$1_{6,2}$			$1_{6,3}$		
		$1_{7,0}$	$1_{7,1}$				$1_{7,2}$			$1_{7,3}$	
$1_{8,0}$				$1_{8,1}$	$1_{8,2}$				$1_{8,3}$		

Figure 1 A (12,3,4) QC-LDPC H-matrix with totally 36 addresses to be stored.

where $y_{u,v}$ can be represented by an $u \times v$ ($0 \leq u \leq j-1$) matrix as shown in Figure 2.

The offset address represented by $1_{r,v}$ is determined as follows:

$$\begin{aligned} & \text{if } (y_{u,v} + c) < p \\ & \quad 1_{r,v} = (y_{u,v} + c) + pv; \\ & \text{else} \\ & \quad 1_{r,v} = (y_{u,v} + c) - p + pv; \end{aligned} \quad (1)$$

where $c = r \bmod p$. For example, $1_{0,1}$ and $1_{5,3}$ in Figure 1 are determined by

$$1_{0,1} = (y_{0,1} + 0) + 3 \times 1 = 4.$$

$$1_{5,3} = (y_{1,3} + 2) - 3 + 3 \times 3 = 9.$$

As a result, an offset address can easily be determined by a simple logic as shown in Figure 3. In (1), the comparison result of $(y_{u,v} + c) < p$ is obtained from a borrow bit. An accumulator of p , instead of a multiplier, can be employed to determine pv as the governing scanning operation accesses addresses in a specific order, and the parameter c is obtained by a modulo- p counter instead of a divider. All the parameters in (1) have sizes of less than Z bits. Therefore, the QC-LDPC block codes can be characterized by a 2-D matrix $[5]\mathbf{Y}$ ($\mathbf{Y} = y_{u,v}$) in which each element is a z -bit ($z \geq \log_2 p$) data representing a shift value in the corresponding identity sub-matrix.

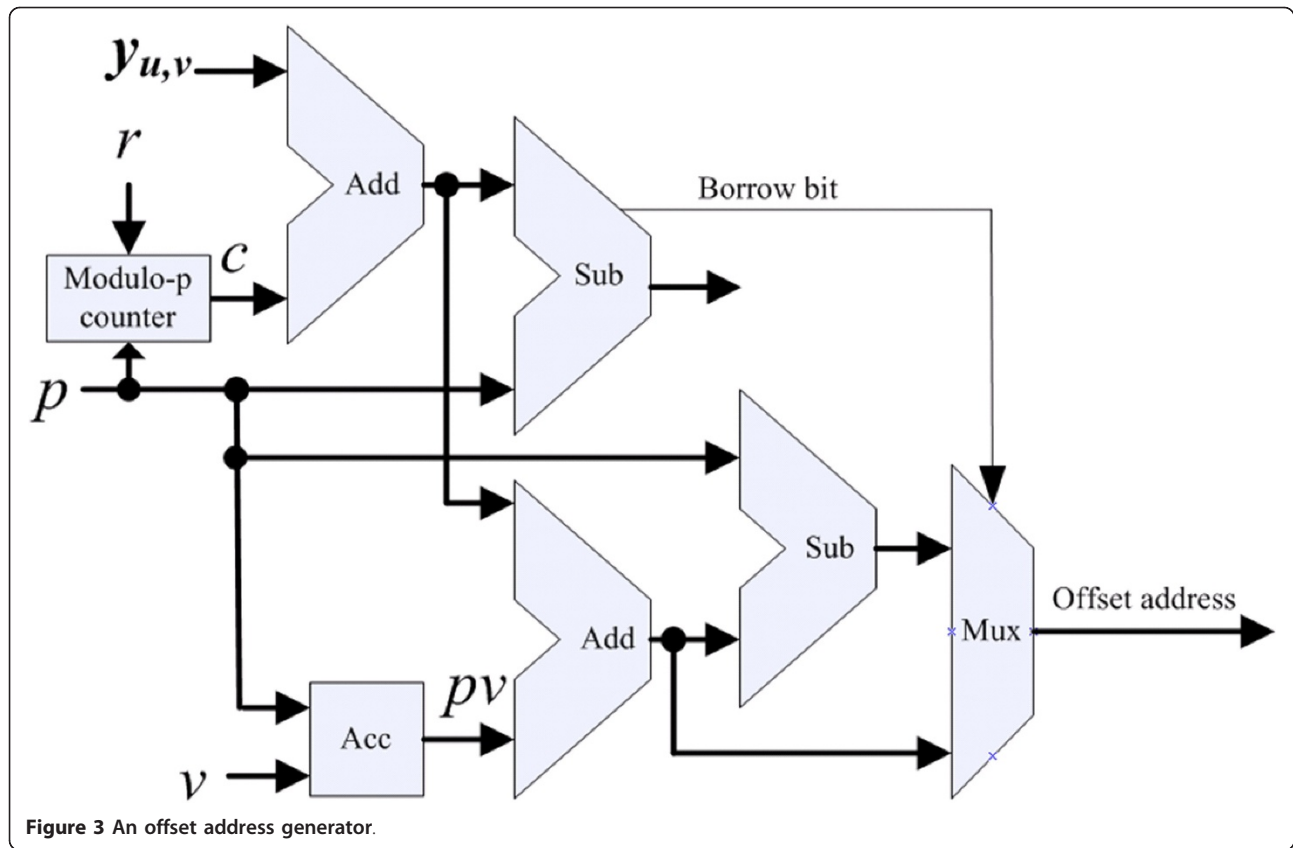
3. Reduction factors in recent memory reduction works

This section surveys recent studies related to the compact storage of a parity-check \mathbf{H} -matrix. As shown in Table 2, the achievements of these studies are evaluated by the reduction factor with respect to the requirements

The figure shows a 4x4 matrix of bits. The top-left corner of the matrix is labeled with $y_{u,v}$ and has two lines extending from it, one labeled u pointing to the first column and one labeled v pointing to the first row. The matrix contains the following bits:

0	1	1	0
1	1	0	1
1	0	1	1

Figure 2 A reduced $j \times k$ matrix \mathbf{Y} for \mathbf{H} -matrix.



of memory bits. An LDPC code is represented by its sparse parity-check matrix of size $M \times N$ and of density [16] defined as $N_e/(M \times N)$ [17] (N_e is the number of ones in the \mathbf{H} -matrix and M is equal to $N-K$). A direct representation of \mathbf{H} exploits its sparseness to record only the non-zero column elements in each row or the non-zero row elements in each column. Hence, $N \times j$ [18] elements or $M \times k$ [5] elements need to be recorded. The LDPC codes designed for the China Mobile Multimedia Broadcasting (CMMB) [5] system are cyclic-shifted with 36 bits every R_p row ($R_p = 18$ in $rate = 1/2$ and $R_p = 9$ in $rate = 3/4$) and the required memory to store all the nonzero elements is $M \times k \times Z$ ($Z \geq \log_2 N$) bits without applying an AGU. With an AGU, the feature of \mathbf{H} is abstracted by two matrices (\mathbf{H}' -matrix and \mathbf{I} -matrix) which generate the actual addresses on the fly. The other memory reduction methods employ only one reduced matrix. These methods are compared with each other by a reduction factor with reference to $M \times k \times Z$. As shown in Table 2, the memory requirements, reduction factors, and application examples of these methods are available.

Due to the limitation of choosing p (p is a prime), we are not able to exactly construct the same codes with the existing approaches. However, to make relatively fair comparisons, some similar codes are constructed to exploit the

memory efficiency of the proposed MWC-OCS LDPC codes. The similar codes are defined as the codes which have similar code lengths, the same code rates, and the same weight k as the compared codes. In terms of the requirement of memory bits shown in Table 3, the state-of-the-art work [5] for CMMB reduced the \mathbf{H} -matrices (rate = 0.5, 0.75) to merely 1728 bits. In our similar codes, 88 and 140 bits are required, respectively, for the described CMMB two-rate \mathbf{H} -matrices. Since these two matrices can be merged into one, 140 bits are required by our similar codes, and hence a 92% (1588/1728) reduction rate is achieved. Furthermore, compared to [2], our similar codes (rate = 0.5) for WLAN and WMAX also achieve a reduction rate of 88%.

Table 3 and Figure 4 also show the synthesis results of our similar codes (synthesized by Synplify Pro 7.2). The code storage (including the cost of addressing the memories and the cost of routing the data) of our similar codes for CMMB require 30 gates, 0.4% of 7 k gates required in [5]. The gain (99.6%) evaluated by the gate count is even larger than the gain (92%) estimated by memory bits. This difference is attributed to the fact that cost of addressing a more compact memory space is much simpler. We also synthesized our similar codes for WLAN and WMAX, the synthesis results show merely 71 and 83 gates are required, respectively. As for this

Table 2 List of works related to non-volatile memory reduction for LDPC codes

Work	Memory requirement	Reduction factor with reference to $M \times k \times Z$	Application examples			
			Code length	Code rate	Memory requirement	Reduction Factor
LDPC for CMMB (all elements) [5]	$M \times k \times Z$	1	9216	1/2	$4608 \times 6 \times 14 = 387072$ bits	1
			9216	3/4	$2304 \times 12 \times 14 = 387072$ bits	1
LDPC for CMMB with AGU [5]	H'	$(Z \times N_p + 2 k \times R_p)/(M \times k \times Z)$	9216	1/2	$1512 + 216 = 1728$ bits	4.46×10^{-3}
	$Z \times N_p$		9216	3/4	$1512 + 216 = 1728$ bits	4.46×10^{-3}
Regular HC-LDPC [17]	$(N_e/N_p) \times \log_2 N_e$	$[\log_2(M \times k)/Z] \times (1/N_p)$	4128	1/2	$(2064 \times 6/18) \times 14 = 9632$ bits	5.98×10^{-2}
TS-LDPC [18]	$(j-1) \times (k-1) \times z$	$[(j-1) \times (k-1)/(M \times k)] \times (z/Z)$	6084	3/4	$3 \times 15 \times 9 = 405$ bits	1.7×10^{-3}
LDPC for WLAN [2]	N.A.	N.A.	1944	1/2	$24 \times 12 \times 7 = 2151$ bits	2.78×10^{-2}
LDPC for WIMAX [2]			2304	1/2	$24 \times 12 \times 7 = 2151$ bits	2.34×10^{-2}
Proposed 1-D MWC-OCS LDPC	$(j + k-1) \times z$	$[(j + k-1)/(M \times k)] \times (z/Z)$	9186 ($p = 1531$)	1/2	$(6 + 3-1) \times 11 = 88$ bits	2.3×10^{-4}
			9228 ($p = 769$)	3/4	$(12 + 3 - 1) \times 10 = 140$ bits	3.6×10^{-4}
			Similar codes for CMMB			
			1992 ($p = 83$)	1/2	$(24 + 12) \times 7 = 252$ bits	9.6×10^{-4}
			2328 ($p = 97$)	1/2	$(24 + 12) \times 7 = 252$ bits	7.5×10^{-4}
			Similar codes for WLAN/WIMAX			

Table 3 Synthesis results of the code storage for the similar codes of CMMB, WLAN, and WMAX

Works		Number of bits	Number of gates
CMMB (rate = 1/2,3/4)	[5]	1728	7 k
	This work (similar code)	140	30
	Gain	92%	99.6%
WLAN (rate = 1/2)	[2]	2151	N.A.
	This work (similar code)	252	71
	Gain	88%	N.A.
WMAX (rate = 1/2)	[2]	2151	N.A.
	This work (similar code)	252	88
	Gain	88%	N.A.

part, the gate count consumed by a rate = 0.5 **H**-matrix is not available in [2].

Compared to the existing approaches, many more **H**-matrices constructed by our approach for diverse communication protocols can be supported within a fixed non-volatile memory space. The extremely low memory requirements of the proposed codes are achieved by a 2-D to 1-D memory space mapping. Consequently, the j

$\times k \times Z$ bits required by a traditional 2-D storage for QC-LDPC codes can be reduced to $(j + k - 1) \times z$ bits, where z is the smallest integer satisfying $z \geq \log_2 p$.

4. Methods to Construct memory-efficient MWC-OCS LDPC codes

A cycle in a Tanner graph is a sequence of connected vertices that starts and ends at the same vertex in the

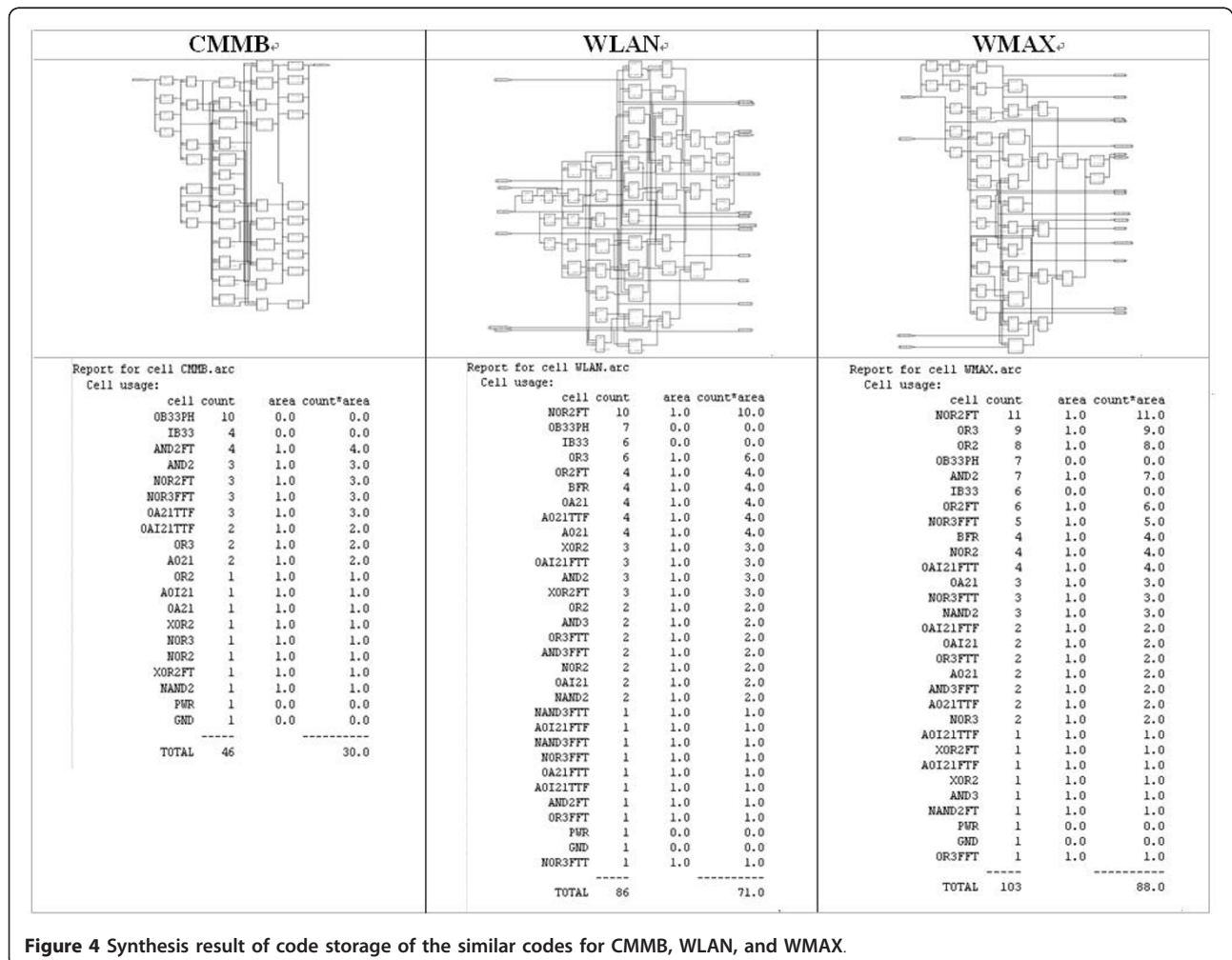


Figure 4 Synthesis result of code storage of the similar codes for CMMB, WLAN, and WMAX.

graph, and which contains other vertices no more than once. To upgrade the performance of LDPC codes, it is necessary to avoid 4-cycles, which is the shortest possible length for a Tanner graph. The girth of an LDPC code is the length of the smallest cycle. Since cycles of short length may degrade the performance of LDPC codes, it is necessary to ensure that the Tanner graph of the LDPC codes is free of cycles with lengths of 4 and hence have girths of at least 6 [13]. In Section 4.1, we introduce how to construct the proposed memory-efficient QC-LDPC codes and prove that no 4-cycles are present. The construction examples and the simulation results are shown at the last two subsections.

4.1. Construction procedure

In this section, we propose a method for constructing the QC-LDPC codes with memory reduction. For clarity of exposition, the MWC-OCS LDPC codes [11] are used as an exemplification. However, for the application of the proposed memory-efficient scheme, only the cyclic parts of the MWC sequences are adopted in the construction procedure.

An element from $GF(p)$ in a $j \times k$ preliminary matrix \mathbf{Y} can be represented as $y_{u,v}$, where the (u, v) th element of \mathbf{Y} can be calculated by Equation (2) with the corresponding values of a_u and b_v , for fixed parameters ϕ and α .

$$y_{u,v} = \alpha \beta^{(a_u + b_v)} + \phi, \quad (2)$$

where $0 \leq u \leq j-1$, $0 \leq v \leq k-1$.

For memory efficiency, two specific sequences $\{a_0, a_1, \dots, a_{j-1}\}$ and $\{b_0, b_1, \dots, b_{k-1}\}$ are constructed by the following procedure.

-First, choose basic parameters j , k , and p (p is a prime) where the $j, k \in \text{integer}$.

-Second, choose initial values a_0 , b_0 , and grid size f where $0 \leq a_m (= a_0 + mf) \leq p-2$ for $m = 0, 1, \dots, j-1$, $1 \leq b_n (= b_0 + nf) \leq p-1$ for $n = 0, 1, \dots, k-1$, and $f \in \{1, 2, \dots, p-1\}$.

-Third, substitute $a_m = a_0 + mf$ and $b_n = b_0 + nf$ into (2). Note that $a_i \neq a_j$ and $b_i \neq b_j$ if $i \neq j$. where $\alpha \in \{1, 2, \dots, p-1\}$, $\phi \in \{0, 1, \dots, p-1\}$, and β is the primitive element of $GF(p)$. Then the following equation is obtained:

$$y_{m,n} = \alpha \beta^{[(a_0 + mf) + (b_0 + nf)]} + \phi \quad (3)$$

The proposed parity check matrix \mathbf{H} , which reduces the non-volatile memory demand, can be represented by a $j \times k$ array of circulant permutation sub-matrices shown as follows:

$$\mathbf{H} = \begin{bmatrix} I(y_{0,0}) & I(y_{0,1}) & \cdots & I(y_{0,k-1}) \\ I(y_{1,0}) & I(y_{1,1}) & \cdots & I(y_{1,k-1}) \\ \vdots & \vdots & \ddots & \vdots \\ I(y_{j-1,0}) & I(y_{j-1,1}) & \cdots & I(y_{j-1,k-1}) \end{bmatrix} \quad (4)$$

where $I(x)$ is a $p \times p$ identity sub-matrix with rows cyclically shifted to the right by x positions. For example, $I(1)$ is the following permutation matrix:

$$I(1) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \quad (5)$$

Hence, the resulting \mathbf{H} , which has j ones in each column and k ones in each row, represents a (j, k) -regular LDPC code (this LDPC code is also an $[N, K]$ regular LDPC code, where $N (= kp)$ is the block length of the MWC-OCS LDPC code and K is the number of message bits).

The size of the parity-check matrix \mathbf{H} is $jp \times kp$. Due to the linear dependence among the rows of \mathbf{H} , it has a code rate of $r = K/N \geq 1-(j/k)$. Actually, since the summation of the p rows of the J th sub-matrices $[I(y_{J,0}) \ I(y_{J,1}) \ \dots \ I(y_{J,k-1})]$ ($0 \leq J \leq j-1$) in (4) are equal to an all-1 vector, there are at least $j-1$ dependent rows in \mathbf{H} . Therefore, the Tanner graph of the resulting LDPC codes is free of cycles with lengths of 4 and hence has a girth of at least 6.

Based on (3), if two sequences a_m and b_n are both ordered incrementally or decrementally by f , an element $y_{s,t}$ will be equal to $y_{m,n}$ as $s+t$ is equal to $m+n$ in the preliminary matrix \mathbf{Y} . In (6), we also verify that if a_m and b_n are ordered in an opposite direction to each other, an element $y_{s,t}$ is still equal to $y_{m,n}$ when $s-t = m-n$. Note that $m \neq s$, $n \neq t$.

$$\begin{aligned} y_{s,t} &= \alpha \beta^{[a_0 + s(+f) + b_0 + t(-f)]} + \phi \\ &= \alpha \beta^{[a_0 + b_0 + (s-t)f]} + \phi \\ &= \alpha \beta^{[a_0 + b_0 + (m-n)f]} + \phi \\ &= \alpha \beta^{[a_0 + m(+f) + b_0 + n(-f)]} + \phi \\ &= y_{m,n} \end{aligned} \quad (6)$$

4.2. Construction examples

According to (3) and (6), two construction examples are shown as follows. The initial values chosen for the required sequences can be different.

Example A: A $[155, 64]$ MWC-OCS LDPC code ($p = 31$)

Let $j = 3$, $k = 5$, $f = 3$. We select $\{a_0, a_1, a_2\} = \{0, 3, 6\}$ and $\{b_0, b_1, b_2, b_3, b_4\} = \{1, 4, 7, 10, 13\}$. By using (3) and (4) with fixed parameter $\alpha = 1$, $\phi = 0$, we can form the following parity-check matrix:

$$\mathbf{H} = \begin{bmatrix} I(3) & I(19) & I(17) & I(25) & I(24) \\ I(19) & I(17) & I(25) & I(24) & I(28) \\ I(17) & I(25) & I(24) & I(28) & I(12) \end{bmatrix}$$

In Example A, two sequences are both ordered increasingly, and the resulting \mathbf{H} -matrix shows the same shift values in the identity sub-matrices from the lower left to the upper right.

Example B: We select $\{a_0, a_1, a_2\} = \{0, 3, 6\}$ and $\{b_0, b_1, b_2, b_3, b_4\} = \{13, 10, 7, 4, 1\}$, and then the following parity-check matrix is formed:

$$\mathbf{H} = \begin{bmatrix} I(24) & I(25) & I(17) & I(19) & I(3) \\ I(28) & I(24) & I(25) & I(17) & I(19) \\ I(12) & I(28) & I(24) & I(25) & I(17) \end{bmatrix}$$

Example B shows that a memory-efficient LDPC codes also can be constructed by two sequences ordered in the opposite direction of each other. The \mathbf{H} -matrix is constructed by $a_m = \{a_0, a_0+f, a_0+2f, \dots, a_0+(j-1)f\}$ and $b_n = \{b_0, b_0-f, b_0-2f, \dots, b_0-(k-1)f\}$. The resulting \mathbf{H} -matrix shows the same shift values in the identity sub-matrices from the upper left to the lower right. Each resulting \mathbf{H} -matrix shown in Examples A and B is a 93×155 matrix and describes a (3, 5)-regular LDPC code with $rate = 64/155 \approx 0.4129$ (by using Gaussian elimination, we know that \mathbf{H} has a rank of 91).

5. 2-D to 1-D mapping and simulation results of memory-efficient MWC-OCS LDPC codes

Due to the memory-efficient property mentioned in Section 4, the corresponding parity-check \mathbf{H} matrices of the proposed MWC-OCS LDPC block exploit superior regularity. This regular code structure enables a mapping of a 2-D matrix into a compact 1-D memory space. As shown in Figure 5, the \mathbf{H} -matrix in Example A is mapped from a 2-D $y_{u,v}$ matrix into a 1-D memory space y_w . As a result, the required memory can be reduced from $j \times k \times z$ bits to $(j+k-1) \times z$ bits, where $w = u + v$. The \mathbf{H} -matrix in Example B is constructed by two sequences ordered in opposite directions of each other. In memory indices transformations, as shown in Figure 6, a 2-D $y_{u,v}$ matrix in Example

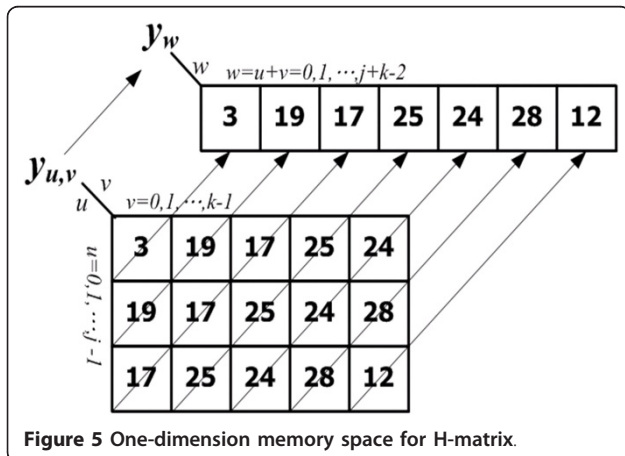


Figure 5 One-dimension memory space for \mathbf{H} -matrix.

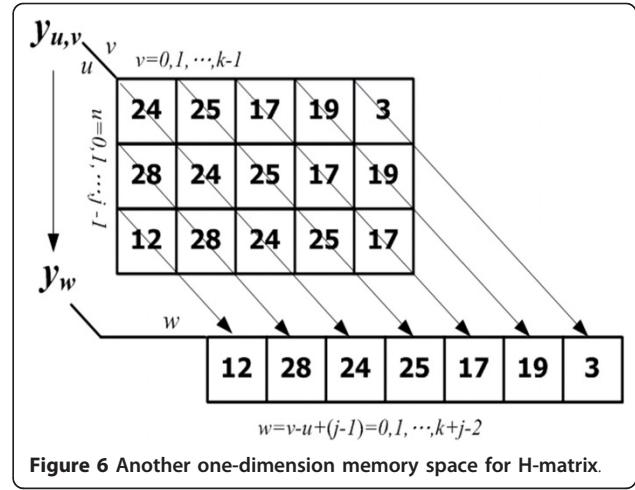


Figure 6 Another one-dimension memory space for \mathbf{H} -matrix.

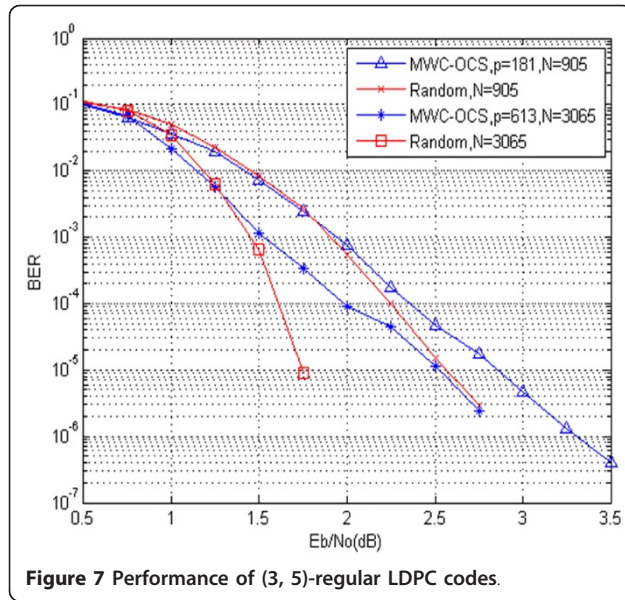
B is mapped into a 1-D memory space y_w where $w = v - u + (j-1)$. No doubt, these two \mathbf{H} -matrices in Examples A and B exhibit the same memory-efficient feature. Therefore, the construction of memory-efficient MWC-OCS LDPC block codes can be conducted as follows:

We construct two specific sequences $\{a_0, a_1, \dots, a_{j-1}\}$ and $\{b_0, b_1, \dots, b_{k-1}\}$, which satisfy the conditions that $0 \leq a_m (= a_0 \pm mf) \leq p-2$ for $m = 0, 1, \dots, j-1$, $1 \leq b_n (= b_0 \pm nf) \leq p-1$ for $n = 0, 1, \dots, k-1$, $f \in \{1, 2, \dots, p-1\}$ and p is an odd prime. Note that $a_i \neq a_j$ and $b_i \neq b_j$ if $i \neq j$. Then the following two cases are able to construct memory-efficient MWC-OCS LDPC codes, which can be mapped from 2-D \mathbf{H} -matrices into 1-D memory spaces.

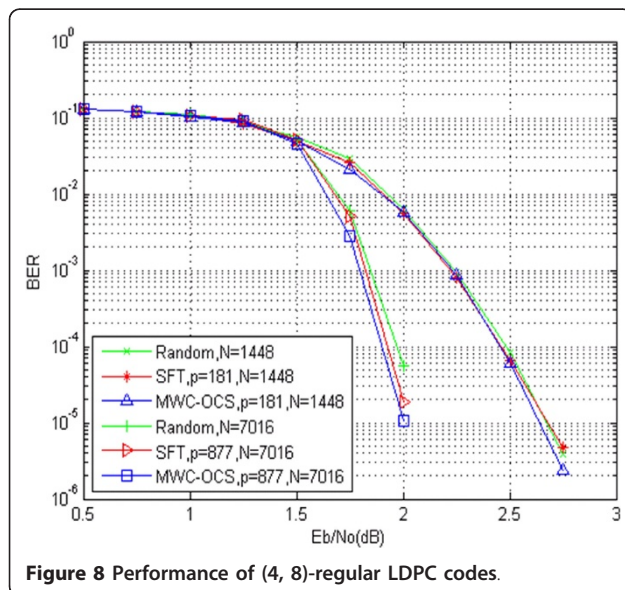
Case A: Two sequences are ordered in the same direction, then the 2-D matrix $y_{u,v}$ is mapped into a 1-D memory space y_w where $w = u + v$.

Case B: Two sequences are ordered in opposite directions of each other, and then the 2-D matrix $y_{u,v}$ is mapped into a 1-D memory space y_w where $w = v - u + (j-1)$.

In addition to the memory-efficient feature, the proposed codes are also required to provide good error performance. To show the simulation results of the error performances which can be achieved by the memory-efficient MWC-OCS LDPC codes, the bit error rates of the proposed codes and the competitive codes are compared via a binary phase-shift keying-modulated additive white Gaussian noise channel with signal-to-noise ratio E_b/N_0 . In all cases, the iterative sum-product algorithm was used for decoding. The proposed (3, 5)-regular memory-efficient MWC-OCS LDPC code in Figure 7 is not as good as the randomly constructed LDPC codes. It shows an error floor which may be caused by their limited minimum distance (for a (j, k) -regular QC-LDPC code, the minimum distance is at most $(j+1)!$ [19]). This performance loss may be attributed to the fact that we have introduced various constraints on the set of code parameters, which influence the performance of belief



propagation decoding. To resolve this problem, several memory-efficient MWC-OCS LDPC codes with column-weight 4 are constructed. As shown in Figure 8, the memory-efficient MWC-OCS LDPC codes perform slightly better than the randomly constructed LDPC codes and Sridhara-Fuja-Tanner (SFT) codes [7]. Figure 9 depicts the performance of high-rate LDPC codes with different constructions. The performances of $rate = 0.75$, (4, 16)-regular LDPC codes with two different block lengths $N = 1648$ and $N = 4016$ are shown. It can be seen that the proposed memory-efficient MWC-OCS LDPC codes outperform the competitive codes under similar code rates and moderate block lengths.



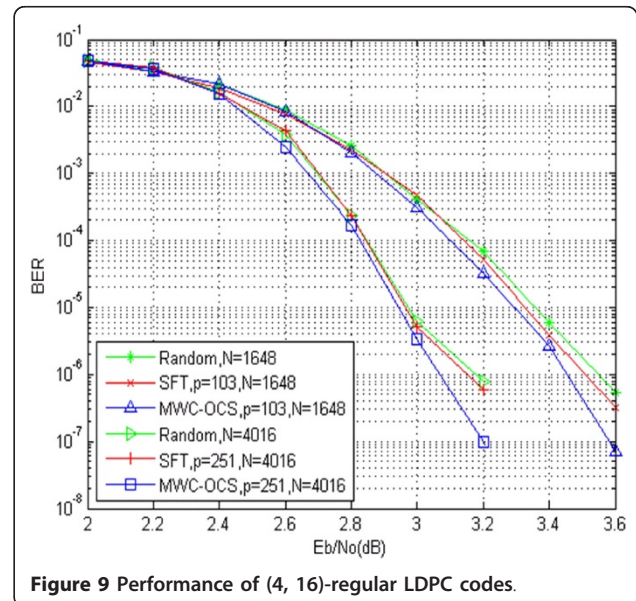
6. A further reduction for multi-rate H-matrices storage

In this section, a design example is demonstrated to show how a further reduction can be achieved after 1-D memory space mapping. As we have mentioned, all possible schemes for further reduction are meaningful as long as they can achieve significant gains, especially when the hardware resource is constrained and the demand for diverse H-matrices increases.

Four diverse H-matrices, which provide comparable error performances in Figures 8 and 9, are compactly saved by the proposed memory reduction scheme. In Figure 10, the four H-matrices are represented by the shift values of the corresponding identity sub-matrices. For example, the element '27' represents an identity sub-matrix $I(27)$ with its shift value parenthesized. The corresponding shift value for each identity sub-matrix is generated via Equation (3) using '3' as a primitive element.

6.1. Preprocess of the elements in 1-D memory spaces

The memory requirement for four H-matrices is reduced to 483 bits after 1-D space mapping. To save several H-matrices in multi-rate communications, a further reduction can be achieved by merging the same shift values with the same memory indices into one before synthesis. That is, except for the 1-D memory space mapping mentioned in Section 5, if the overlapped elements in diverse H-matrices can be preprocessed, a better optimization can be achieved in our synthesis result. In Figure 11, each H-matrix has been mapped into a 1-D memory space. We find that these 1-D memory spaces have the same elements in specific



H1 p=181	27	81	62	5	15	45	135	43
	9	27	81	62	5	15	45	135
	3	9	27	81	62	5	15	45
	1	3	9	27	81	62	5	15

H2 P=877	27	81	243	729	433	422	389	290
	9	27	81	243	729	433	422	389
	3	9	27	81	243	729	433	422
	1	3	9	27	81	243	729	433

H3 P=103	27	81	37	8	24	72	10	30	90	64	89	61	80	34	102	100
	9	27	81	37	8	24	72	10	30	90	64	89	61	80	34	102
	3	9	27	81	37	8	24	72	10	30	90	64	89	61	80	34
	1	3	9	27	81	37	8	24	72	10	30	90	64	89	61	80

H4 P=251	27	81	243	227	179	35	105	64	192	74	222	164	241	221	161	232
	9	27	81	243	227	179	35	105	64	192	74	222	164	241	221	161
	3	9	27	81	243	227	179	35	105	64	192	74	222	164	241	221
	1	3	9	27	81	243	227	179	35	105	64	192	74	222	164	241


Figure 10 Four diverse H-matrices for a design example.

memory indices. The required memory can be more compact after all the overlapped elements are merged into one.

6.2. Design architecture

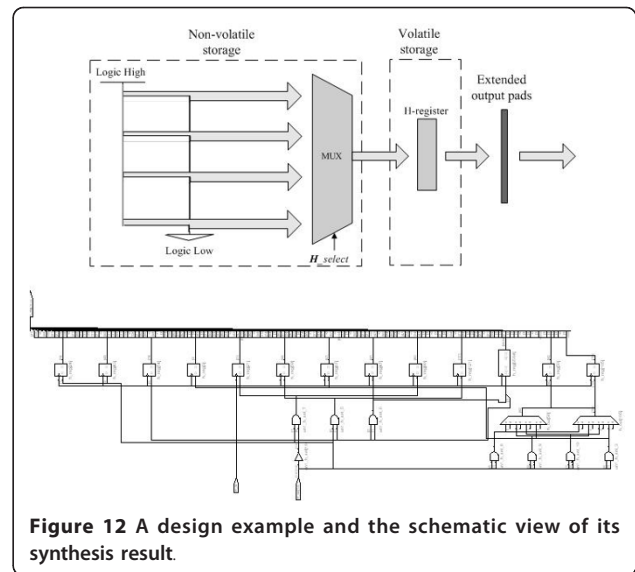
The architecture of the design example is shown in Figure 12. An H-register, employed for saving a specific H-matrix selected from the four H-matrices, prevents accessing of the non-volatile memory frequently during the communication time. Therefore, the implementation can be divided into two parts, namely non-volatile storage and volatile storage. In Figure 12, the non-volatile part is implemented by sticking the inputs of the 4-input multiplexer to logic high or logic low, instead of using a real ROM block. Before a session of communication commences, the input signal H_select selects a specific 1-D memory space and loads its content into an H-register. During communication, only the H-register needs to be accessed to determine the message addresses, and the non-volatile memory access is thus prevented. In a session of communication,

1	3	9	27	81	37	8	24	72	10	30	90	64	89	61	80	34	102	100
1	3	9	27	81	243	227	179	35	105	64	192	74	222	164	241	221	161	232



					62	5	15	45	135	43								
						729	433	422	389	290								
					37	111	24	72	10	30	90	64	89	61	80	34	102	100
1	3	9	27	81	243	227	179	35	105	64	192	74	222	164	241	221	161	232

Figure 11 Preprocess of the same shift values.



all the shift values can be spanned through the content of the H-register. As shown in Table 4, four diverse H-matrices which provide relatively good error performances are included in the H-library. Since only one H-matrix is required for error correction in a specific session of communication, it is not necessary for all the four H-matrices to be loaded once at a time.

6.3. Synthesis results

Before synthesizing the design example, by applying the proposed 2-D to 1-D memory space mapping scheme, the total non-volatile memory demand is reduced to 483 bits, and the corresponding reduction factor is only 6.94×10^{-4} . A further optimization is achieved by a reduction factor of 5.02×10^{-4} after preprocessing the overlapped elements before synthesis. The synthesis report in the last line of Table 4 shows that addressing the memories and routing the data require 13 gates, 152 memory-less and gate-free output pads, and a 16-DFF H-register which spans all the shift values. The schematic view of the synthesis result is also shown in Figure 12. During a communication session, all the shift values of a specific H can be retrieved through the 152-bit extended output pads. These pads send out fixed logic levels mixed with the output of the H-register. Therefore, in a session of communication, only the 152-bit (maximum size of the four 1-D memory spaces) output data spanned by the H-register is accessed for fast retrieval of the shift values. The non-volatile part will not be accessed until a new session of communication using another H-matrix begins.

7. Conclusion

Memory-efficient MWC-OCS LDPC codes were proposed to reduce the non-volatile memory demand for

Table 4 1-D Memory mapping and synthesis result of design example

H-matrix	Type	Non-zero elements storage ($M \times k \times Z$)	Multiple constructions			1-D memory mapping																																																									
			(N, j, k)	Rate	p	$(j + k - 1)z$	Reduction factor																																																								
H1	MWC-OCS	$724 \times 8 \times 11 = 63712$ bits	(1448,4,8)	0.5A	181	$11 \times 8 = 88$ bits	1.38×10^{-3}																																																								
H2	MWC-OCS	$3508 \times 8 \times 13 = 364832$ bits	(7016,4,8)	0.5B	877	$11 \times 10 = 110$ bits	3.01×10^{-4}																																																								
H3	MWC-OCS	$412 \times 16 \times 11 = 72512$ bits	(1648,4,16)	0.75A	103	$19 \times 7 = 133$ bits	1.83×10^{-3}																																																								
H4	MWC-OCS	$1004 \times 16 \times 12 = 192768$ bits	(4016,4,16)	0.75B	251	$19 \times 8 = 152$ bits	7.88×10^{-4}																																																								
H-library	MWC-OCS	693824 bits				483 bits	6.96×10^{-4}																																																								
Preprocess	MWC-OCS	693824 bits				348 bits	5.02×10^{-4}																																																								
Synthesis results	MWC-OCS H-library	Report for cell H_1lib.aec Cell usage: <table><tr><th>cell count</th><th>area</th><th>count</th><th>area</th></tr><tr><td>OB33PH 152</td><td>0.0</td><td>0.0</td><td></td></tr><tr><td>DFF 16</td><td>1.0</td><td>16.0</td><td></td></tr><tr><td>INV 7</td><td>1.0</td><td>7.0</td><td></td></tr><tr><td>NOR2FT 2</td><td>1.0</td><td>2.0</td><td></td></tr><tr><td>IB33 2</td><td>0.0</td><td>0.0</td><td></td></tr><tr><td>AND2 1</td><td>1.0</td><td>1.0</td><td></td></tr><tr><td>NOR2 1</td><td>1.0</td><td>1.0</td><td></td></tr><tr><td>XOR2FT 1</td><td>1.0</td><td>1.0</td><td></td></tr><tr><td>FOR 1</td><td>0.0</td><td>0.0</td><td></td></tr><tr><td>GND 1</td><td>0.0</td><td>0.0</td><td></td></tr><tr><td>GL33 1</td><td>0.0</td><td>0.0</td><td></td></tr><tr><td>FFR 1</td><td>1.0</td><td>1.0</td><td></td></tr><tr><td>TOTAL 186</td><td></td><td></td><td>29.0</td></tr></table> Found clock H_1libH_1d with period 1000.00ns				cell count	area	count	area	OB33PH 152	0.0	0.0		DFF 16	1.0	16.0		INV 7	1.0	7.0		NOR2FT 2	1.0	2.0		IB33 2	0.0	0.0		AND2 1	1.0	1.0		NOR2 1	1.0	1.0		XOR2FT 1	1.0	1.0		FOR 1	0.0	0.0		GND 1	0.0	0.0		GL33 1	0.0	0.0		FFR 1	1.0	1.0		TOTAL 186			29.0	152 output pads (memory less and gate free) 13 gates (non-volatile) 16 DFFs (volatile)	
cell count	area	count	area																																																												
OB33PH 152	0.0	0.0																																																													
DFF 16	1.0	16.0																																																													
INV 7	1.0	7.0																																																													
NOR2FT 2	1.0	2.0																																																													
IB33 2	0.0	0.0																																																													
AND2 1	1.0	1.0																																																													
NOR2 1	1.0	1.0																																																													
XOR2FT 1	1.0	1.0																																																													
FOR 1	0.0	0.0																																																													
GND 1	0.0	0.0																																																													
GL33 1	0.0	0.0																																																													
FFR 1	1.0	1.0																																																													
TOTAL 186			29.0																																																												

H-matrix storage. The described similar codes outperform other recent approaches with huge memory-reduction rates. Compared to CMMB and WIMAX/WLAN, our similar codes have, respectively, achieved 92 and 88% reduction rates in terms of the requirement of memory bits. In the synthesis result of our similar code storage for CMMB, the gain (99.6%) evaluated by gate count is even larger than the gain (92%) estimated by memory bits. In addition, the proposed codes also show relatively good error performances comparable with competitive codes in column-weight 4 constructions. Furthermore, a design example was also synthesized to be every compact for multi-rate H-matrices storage. In implementing wireless applications for multiple communication protocols within a fixed size of memory space, our approach is of good worth. As we have mentioned, even when the huge gain achieved is only a small fraction of the overall decoder, it cannot be considered negligible as the demand for diverse H-matrices storage of multiple communication protocols increases. Especially, for a tiny device with a very limited size of memory space, applying the proposed approach can enhance the capability for supporting many more error correction functions.

Acknowledgements

This research was partially supported by the National Science Council in Taiwan (Grant No.NSC98-2221-E-168-004-MY3).

Author details

¹Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan, R.O.C. ²Department of Electro-Optical Engineering, Kun Shan University, Tainan 710, Taiwan, Republic of China. ³Chung-Shan Institute of Science and Technology

Competing interests

The authors declare that they have no competing interests.

Received: 8 February 2012 Accepted: 8 June 2012

Published: 8 June 2012

References

1. D MacKay, R Neal, Near Shannon limit performance of low-density parity-check codes. *Electron Lett.* **32**, 1645–1646 (1996). doi:10.1049/el:19961141
2. E Amador, R Pacalet, V Rezaei, Optimum LDPC decoder: a memory architecture problem, in *Proceedings of the Design Automation Conference*, (San Francisco, CA, USA, 2009), pp. 891–896
3. Y Tian, X Zhang, Z Lai, A LDPC decoder with all single port memories, in *Proceedings of the Intelligent Computing and Intelligent Systems, Shanghai, China*. **3**, 547–550 (2009)
4. F Kienle, T Brack, N When, A synthesizable IP Core for DVB-S2 LDPC code decoding, in *Proceedings of the Design Automation and Test in Europe 2005*, Munich, Germany, **3**, 100–105 (2005)
5. S Lee, J Park, K Chung, Memory efficient multi-rate regular LDPC decoder for CMMB. *IEEE Trans Consum Electron.* **55**(4), 1866–1874 (2008)
6. A Prabhakar, K Narayanan, Pseudorandom construction of low density parity check codes using linear congruential sequences. *IEEE Trans Commun.* **50**(9), 1389–1396 (2002). doi:10.1109/TCOMM.2002.802537
7. R Tanner, D Sridhara, A Sridharan, T Fuja, D Costello, LDPC block and convolutional codes based on circulant matrices. *IEEE Trans Inf Theory.* **50**(12), 2966–2984 (2004). doi:10.1109/TIT.2004.838370
8. M Fossorier, Quasi-cyclic low density parity check codes from circulant permutation matrices. *IEEE Trans Inf Theory.* **50**, 1788–1794 (2004). doi:10.1109/TIT.2004.831841
9. Z Kostic, E Tittlebaum, The design and performance analysis for several new classes of codes for optical synchronous CDMA and for arbitrary-medium time-hopping synchronous CDMA communication systems. *IEEE Trans Commun.* **42**, 2608–2617 (1994). doi:10.1109/26.310621
10. L Chen, J Xu, I Djurdjevic, S Lin, Near-Shannon-limit quasi-cyclic low-density parity-check codes. *IEEE Trans Commun.* **52**(7), 1038–1042 (2004). doi:10.1109/TCOMM.2004.831353
11. J Huang, C Yang, C Huang, On analyzing quasi-cyclic LDPC codes over modified Welch-Costas-coded optical CDMA system. *IEEE J Lightw Technol.* **27**(12), 2150–2158 (2009)
12. C Yang, Optical CDMA passive optical network using prime code with interference elimination. *IEEE Photon Technol Lett.* **19**, 516–518 (2007)
13. J Huang, C Huang, C Yang, Construction of one-coincidence sequence quasi-cycle LDPC codes of large Girth, in *IEEE Trans Inf Theory*, vol. 58. (LDPC Decoder implementation, 2012)(3), pp. 1825–1836http://cwe.ccsds.org. Accessed 9 April 2012
14. High throughput low power decoder architectures for low density parity check codeshttp://repository.tamu.edu. Accessed 9 April 2012
15. S Sandberg, Improved design of unequal error protection LDPC codes. *EURASIP J Wirel Commun Netw.* **2010**. doi:10.1155/2010/423989

16. F Verdier, D Declercq, A low-cost parallel scalable FPGA architecture for regular and irregular LDPC decoding. *IEEE Trans Commun.* **54**(9), 1215–1223 (2006)
17. J Moura, J Lu, H Zhang, Structured low density parity check decoding. *IEEE Signal Process Mag.* **21**, 42–55 (2004). doi:10.1109/MSP.2004.1267048
18. D MacKay, M Davey, Evaluation of Gallager codes for short block length and high rate applications, in *Proceedings of the IMA Workshop Codes, Systems and Graphical Models*, Minneapolis, MN, (1999)
19. Y Dai, N Chen, Z Yan, Memory efficient decoder architecture for quasi-cyclic LDPC codes. *IEEE Trans Circ Syst.* **55**(9), 2898–2911 (2008)

doi:10.1186/1687-1499-2012-191

Cite this article as: Young et al.: Non-volatile memory reduction based on 1-D memory space mapping of a specific set of QC-LDPC codes. *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:191.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Immediate publication on acceptance
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com